IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 25, 2020, accepted June 1, 2020, date of publication June 3, 2020, date of current version June 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2999817

Comparing Admission Control Architectures for Real-Time Ethernet

INÉS ÁLVAREZ[1], (Student Member, IEEE), LUÍS MOUTINHO[2,3], (Member, IEEE), PAULO PEDREIRAS[2,4], (Senior Member, IEEE), DANIEL BUJOSA[5], JULIÁN PROENZA[1], (Senior Member, IEEE), AND LUIS ALMEIDA[6,7], (Senior Member, IEEE)

[1]Department of Mathematics and Informatics, University of the Balearic Islands (UIB), 07122 Palma de Mallorca, Spain
[2]Instituto de Telecomunicações (IT), 3810-193 Aveiro, Portugal
[3]Escola Superior de Tecnologia e Gestão de Águeda (ESTGA), 3750-127 Águeda, Portugal
[4]Department of Electronics, Telecommunications and Informatics, University of Aveiro (UA), 3810-193 Aveiro, Portugal
[5]School of Innovation, Design and Engineering, Mälardalen University, 722 20 Västerås, Sweden
[6]Electrical and Computer Engineering Department, Faculty of Engineering, University of Porto (FEUP), 4200-465 Porto, Portugal
[7]Research Centre in Real-Time and Embedded Computing Systems (CISTER), 4249-015 Porto, Portugal

Corresponding author: Inés Álvarez (ines.alvarez@uib.es)

This work was supported in part by the Spanish Agencia Estatal de Investigación (AEI), in part by the Fondo Europeo de Desarrollo Regional (FEDER) [AEI/FEDER, Unión Europea (UE)] under Grant TEC2015-70313-R, in part by the European Regional Development Fund (FEDER) through the Operational Programme for Competitivity and the Internationalization of Portugal 2020 Partnership Agreement (PRODUTECH-SIF) under Grant POCI-01-0247-FEDER-024541, and in part by the Research Centre Instituto de Telecomunicações under Grant UID/EEA/50008/2013.

ABSTRACT Industry 4.0 and Autonomous Driving are emerging resource-intensive distributed application domains that deal with open and evolving environments. These systems are subject to stringent resource, timing, and other non-functional constraints, as well as frequent reconfiguration. Thus, real-time behavior must not preclude operational flexibility. This combination is motivating ongoing efforts within the Time Sensitive Networking (TSN) standardization committee to define admission control mechanisms for Ethernet. Existing mechanisms in TSN, like those of AVB, its predecessor, follow a distributed architecture that favors scalability. Conversely, the new mechanisms envisaged for TSN (IEEE 802.1Qcc) follow a (partially) centralized architecture, favoring short reconfiguration latency. This paper shows the first quantitative comparison between distributed and centralized admission control architectures concerning reconfiguration latency. Here, we compare AVB against a dynamic real-time reconfigurable Ethernet technology with centralized management, namely HaRTES. Our experiments show a significantly lower latency using the centralized architecture. We also observe the dependence of the distributed architecture in the end nodes' performance and the benefit of having a protected channel for the admission control transactions.

INDEX TERMS Ethernet, TSN, AVB, HaRTES, real-time communication, dynamic reconfiguration, admission control protocol.

I. INTRODUCTION

Adaptive systems are increasingly getting the attention of industry and academia, as there is a high number of applications that could benefit from their characteristics [1], [2]. An adaptive system can modify its behavior to respond to changes in the environment in which it operates, or in the system itself. Such systems commonly have *real-time (RT)* requirements imposed by their interaction with the environment, having to produce a correct output within a bounded time that depends on the dynamics of the system.

For a system to be adaptive, it must be *flexible* at every level of its architecture, including the network. Specifically, we differentiate two types of flexibility at the network level: (i) *real-time flexibility*, *i.e.*, the capacity of the network to transmit traffic with different real-time characteristics (*e.g.*, *time-triggered* and *event-triggered*, *hard*, *soft* and *non-real-time (NRT)*); and (ii) *operational flexibility*, *i.e.*, the ability of the network to allow modifications to the traffic requirements, at runtime, to provide adequate communication as the operational requirements of the system evolve.

The associate editor coordinating the review of this manuscript and approving it for publication was Cong Pu.

Concerning communication technologies, there is a growing interest in using Ethernet in real-time domains that are now exploring adaptivity, such as automotive [3], industrial automation [4], and energy distribution [5]. Ethernet's success is related to its low cost, high bandwidth, compatibility with IP-based networks, and high scalability. However, Ethernet was not originally designed to support adaptive systems with real-time guarantees. Thus, many protocols have been proposed to provide Ethernet with those features, *e.g.*, the industry-driven *Time-Triggered Ethernet* [6] protocol and the academic *Dynamic-TDMA* [7] protocol. Nevertheless, the literature shows that protocols providing RT flexibility usually lack operational flexibility and vice-versa.

One domain where both types of flexibility have coexisted is multimedia. In this context, IEEE created the *Audio Video Bridging (AVB)* Task Group [8] to provide Ethernet with both types of flexibility, resulting in a set of four standards, commonly referred to as AVB standards. AVB supports two different classes of soft event-triggered RT traffic and the management of traffic at runtime. The management is done via the Stream Reservation Protocol (SRP) that incorporates a fully distributed *admission control (AC)* that accepts new flows when there are enough network resources to guarantee their requirements.

Eventually, the AVB Task Group was renamed Time-Sensitive Networking (TSN) Task Group to reflect an expanded scope, now including the transmission of low-latency and reliable traffic, *e.g.*, control traffic found in automotive and industrial applications. The AVB mechanisms for handling event-triggered traffic were integrated into TSN, and several enhancements were introduced, *e.g.*, IEEE 802.1Q{av,cc,ch,ci}. Specifically, IEEE 802.1Qcc [9] extends the SRP capabilities in AVB with two new network management approaches: (i) a *fully centralized* model, where end-systems report their stream requirements to a central management entity that then configures all devices accordingly; and (ii) a *centralized network/distributed user* model, where end-systems send their requests to a close edge bridge that forwards the requests to the central management entity.

The new network management methods in IEEE 802.1Qcc shift from the fully distributed AC of SRP in AVB to a centralized or hybrid AC architecture aiming at real-time handling of change requests, *i.e.*, the transmission of requests, the evaluation of resources, and the propagation and enforcement of the results must be deterministic and completed within a strictly bounded time. An initial comparison of the hybrid and distributed management methods is shown in [10] using simulation and focused on the configuration of time-triggered Time-Aware Shapers (IEEE 802.1Qbv) aiming at maximizing scheduled traffic streams.

Nevertheless, to the best of the authors' knowledge, there is yet no experimental study specifically aimed at characterizing the benefits of moving from a distributed to a centralized AC architecture with existing equipment and focused on event-triggered traffic. This is, thus, the primary motivation of this work.

Since IEEE 802.1Qcc in TSN is not yet available in practice, we used another Ethernet-based protocol, namely Hard Real-Time Ethernet Switch (HaRTES) [11], that uses a centralized AC, to be compared against the distributed AC of AVB, focusing on the architectural differences of the respective AC processes. HaRTES is an academic implementation on switched Ethernet of the *Flexible Time-Triggered (FTT)* [12] paradigm that was devised to provide native real-time and operational flexibility.

The comparison is based on an experimental evaluation of the impact of the AC architectures on the AC latency. We claim that this comparison is meaningful for the ongoing development of network management protocols for reconfigurable distributed real-time systems because: (i) it provides absolute performance values of the AC latency achieved with existing technologies; and (ii) it provides a brief sensitive study of how AC processes based on opposite architectural paradigms depend on relevant network and system parameters.

The paper is organized as follows. The following section presents an overview of related work. Section III presents a brief description of both protocols and a qualitative comparison that exposes relevant intrinsic properties. Their AC processes are detailed in Section IV with a brief reference to the corresponding schedulability tests in Section V. The experimental results of AC latency are shown and discussed in Section VI. Finally, Section VII concludes the paper.

II. RELATED WORK

Supporting real-time network reconfiguration is gaining relevance in domains such as the automotive or the automation industry.

Previous works proposed using Ethernet AVB for automotive systems [13], taking advantage of its fully distributed AC process for resource reservations, namely SRP. More recently, Ethernet Time-Sensitive Networking (TSN) [14] has also been proposed for the same scope, with centralized and hybrid dynamic network management (IEEE 802.1Qcc).

However, as referred in Section I, many of the operational mechanisms of the network management standard are still to be defined and, thus, practical implementations of the centralized architectures are lacking. Furthermore, at the moment of writing this paper, there is no available YANG model to configure the reservations of event-triggered traffic using the centralized architectures proposed in TSN and, thus, the distributed version used in AVB is still the only one available in practice in the TSN (AVB) framework.

Therefore, in this work, for the centralized AC architecture counterpart we use HaRTES as a real-time reconfigurable Ethernet protocol. We constrain the protocol to event-triggered traffic, only, for the sake of direct comparison with AVB. For this reason, we also discard other real-time reconfigurable protocols that aim at time-triggered traffic, only, such as Dynamic-TDMA [7]. Also note that using SRP, from AVB, subsumes previous work on real-time

reconfigurable Ethernet protocols that used a similar distributed resource reservation approach, such as Ethereal [15].

The comparison between the two AC architectural models, using AVB and HaRTES, is based on an automotive-inspired use case, adapted from Meyer *et al.* [16] and Ko *et al.* [17]. This and other related literature [18]–[20] addresses the timing analysis of the protocols, which goes beyond the scope of this paper. Nevertheless, these works clearly show the relevance of considering a multi-hop topology in automotive systems, prompting us to do similarly.

A preliminary comparison of the AC processes of AVB and HaRTES was presented in [21]. This comparison covered aspects such as reliability, flexibility, and performance, but the analysis was qualitative, only, and non-exhaustive. In the current work, we experimentally measure the time required by each protocol to carry out the AC process under different scenarios that highlight the impact of the respective AC architecture. To the best of the authors' knowledge, there are no other works devoted to comparing the timeliness of the AC processes of real-time reconfigurable Ethernet protocols using opposite architectural paradigms, namely distributed (AVB) and centralized (HaRTES).

We believe this novel comparison sheds light on the potential gains achievable in TSN by moving from the fully distributed to centralized dynamic network management architectures, for event-triggered traffic too. Our work is complementary to the work in [10] that assesses, using simulation, only, the signalling of the AC to reconfigure TSN Time-Aware Shapers (time-triggered), with the hybrid and distributed management architectures. Conversely, we assess the AC process as a whole with existing equipment. Moreover, the work in [10] does not evaluate how the different parameters of the network impact the admission control process, which is an important part of our work.

III. OVERVIEW OF THE BASE PROTOCOLS

This section provides a brief introduction to HaRTES and AVB to better understand their nature, operation, and AC protocols.

A. OVERVIEW OF AUDIO VIDEO BRIDGING

The AVB Task Group from the IEEE proposed a set of standards to provide standard Ethernet with: (i) support for soft RT traffic; (ii) shaping of the RT traffic; (iii) resource reservation for the RT traffic; (iv) global time synchronisation; (v) on-line management of the parameters of the RT traffic and (vi) seamless integration of NRT Ethernet end-stations (nodes). This set of standards is commonly referred to as AVB.

AVB is based on bridged networks, through which end-stations exchange information. The real-time communication is carried out through virtual communication channels called streams (or flows) and follows a publisher-subscriber model. End stations can act as talkers (publishers) or listeners (subscribers) of streams.

AVB is composed of three technical standards and one profile, which puts those standards together, to provide the above services. The technical standards are the IEEE 802.1AS: Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks [22]; IEEE 802.1Qav: Forwarding and Queuing Enhancements for Time-Sensitive Streams [23] and IEEE 802.1Qat: SRP [24]. The profile that puts them all together is the IEEE 802.1BA-2011: Audio Video Bridging AVB Systems [25].

The IEEE 802.1AS is a profile of the *Precision Time Protocol (PTP)* [26]. PTP provides global clock synchronization, *i.e.*, it provides a common notion of time to all the bridges and end-stations in the network.

The IEEE 802.1Qav standardises the *Credit-Based Shaper (CBS)*, which provides soft RT guarantees for two different classes of traffic, named A and B. FIGURE 1 shows the operation of the mechanism. Each traffic class has a given credit assigned, which decreases whenever a frame that belongs to the class is transmitted. Once the credit is exhausted (negative), the traffic of that class must wait for transmission and its credit increases. Meanwhile, the traffic from the other class can be transmitted, as long as its credit is positive. This mechanism prevents starvation and bounds the end-to-end delay of the soft RT traffic.

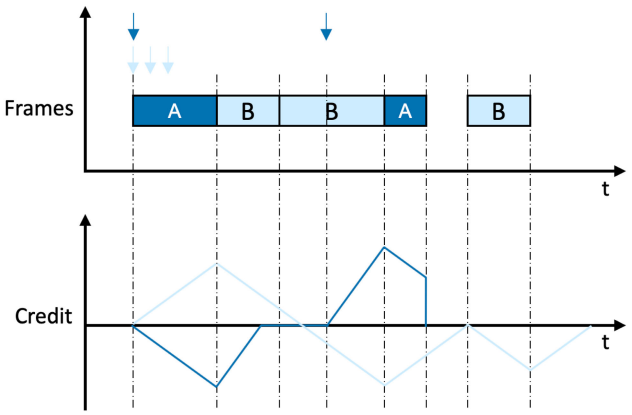

FIGURE 1. Example of the operation of the Credit Based Shaper, reproduced as in [27].

Class A traffic has higher priority and shorter end-to-end delays than Class B traffic. Traffic that does not belong to class A or B is transmitted in the background, on a best-effort approach. This approach allows transparently attaching non-AVB Ethernet end-stations.

The use of CBS by itself is not enough to bound the end-to-end delay of communications. To do so, AVB relies on resource reservations, standardized in the IEEE 802.1Qat SRP. SRP allows talkers to register the attributes of the streams that they wish to transmit and allows listeners to bind to the streams they want to receive. Devices on the stream's path check if the available resources are enough to satisfy the requirements and, if so, lock them, thus assuring that the latency of registered traffic is bounded.

B. OVERVIEW OF HaRTES

The HaRTES switching platform [11] implements the FTT [12] paradigm, providing: (i) support for synchronous (time-triggered) and asynchronous (event-triggered) traffic; (ii) hierarchical, server-based traffic scheduling for the asynchronous traffic; (iii) online stream management without service disruption; (iv) seamless integration of standard Ethernet nodes; and (v) traffic policing and confinement mechanisms.

The FTT paradigm follows a master/multi-slave architecture, in which the master node, implemented as a logical component of the switch, has a set of databases that contain all the attributes related to the message set, communication links, and topology. The master organizes communications cyclically with a fixed-duration time interval called elementary cycle (EC) (FIGURE 2).

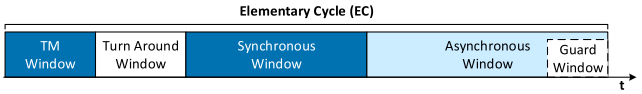

FIGURE 2. Elementary Cycle in HaRTES.

Each EC starts with the broadcast (confined to each switch) of the trigger message (TM) that synchronizes nodes and delivers the schedule of time-triggered traffic for that EC according to a given scheduling policy (*e.g.*, Rate-Monotonic, Earliest Deadline First, Hierarchical Scheduling). These schedules are built assuring that the scheduled traffic fits in the respective EC, thus preventing traffic accumulation inside the switch from one EC to the next. Consequently, at the EC timing scale, the traffic scheduling follows the policy used by the master, irrespective of the policy of internal switch queues. Within the EC, HaRTES-compliant nodes decode the received TM during the turn-around window (TAW) and immediately transmit, within the synchronous window (SW), the scheduled message(s) for which they are the producer(s). The switch then forwards the received messages to the correct egress port(s). By dynamically building the schedule, EC by EC, HaRTES can adapt to updates to the message set without service disruption, thus providing real-time operational flexibility. As the focus of this paper is on the asynchronous traffic, no further details on the synchronous services are provided. For more details, consult [11].

The asynchronous window (AW) is devoted to event-triggered traffic, which is triggered autonomously by nodes, without the intervention of the master. Once the asynchronous traffic is received by the switch, it is queued in dedicated memory pools and verified against the attributes registered on the master's databases. Messages that violate the respective reservation, *e.g.*, maximum message size or minimum inter-arrival time, are discarded or reshaped. On each egress port, a configurable hierarchical traffic scheduling framework (FIGURE 3) manages and dispatches queued messages. Under this framework, the available bandwidth of the AW is divided among a hierarchical tree of traffic shapers, aka *scheduling servers*, each employing a specific scheduling algorithm to regulate the access of its children to the

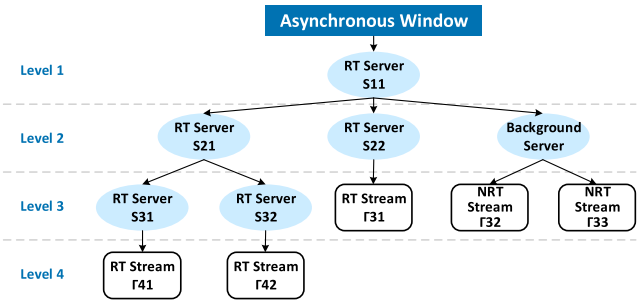

FIGURE 3. Example hierarchy for event-triggered traffic in HaRTES.

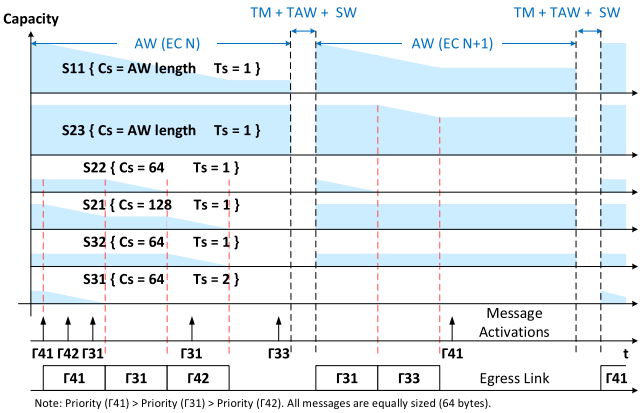

FIGURE 4. Operation of the hierarchical framework based on Deferrable Servers.

bandwidth it provides. The hierarchy can be freely configured, being possible to attach several servers and traffic streams as children of a given server, thus extending in depth as needed. Hierarchical scheduling allows reserving bandwidth with desired timing parameters, bounding mutual interference across streams, favoring system predictability, and analysability [28] and, last but not least, supporting component-based design methodologies.

HaRTES's servers currently employ the *deferrable server (DS)* [29] algorithm, a fixed-priority scheduling policy based on capacity consumption and replenishment. Each server is characterized by a capacity C_s, in bytes, and a replenishment period T_s, specified as multiple of the EC length. The operation of DS, considering the hierarchy example of FIGURE 3, is shown in FIGURE 4. Queued messages are served following their configured priority level and may be sent at any time as long as there is enough capacity available in the associated server and all its parents. As messages are transmitted, their size is subtracted from the available capacity of all involved servers along the hierarchy. In the example, message Γ41 is the first one to arrive and is immediately transmitted. The associated servers $S11$, $S21$ and $S31$ have its capacity decreased by the message size. While message Γ41 is transmitted, messages Γ42 and Γ31 arrive. Γ31 is transmitted first because it has higher priority. In result of its transmission servers $S22$ and $S11$ have its capacity decreased. Then, message Γ42 is transmitted and consequently $S32$, $S21$, and $S11$ have its capacity decreased. The capacity of all

servers is replenished at the beginning of the following EC, except for $S31$, which has a replenishment period of 2 ECs. The figure also shows that messages that are activated close to the end of the EC and that would cause an overrun have its transmission postponed to the following EC, which is the case of $\Gamma 33$.

Finally, NRT traffic is also supported and conveyed in the asynchronous window (AW). A background server is created by default for this purpose, and traffic not associated with a reservation is automatically diverted to this server. Therefore, from the NRT nodes point of view, a HaRTES network behaves as a normal Ethernet network, except that the bandwidth is reduced, proportionally to the existing RT reservations. A guard window (GW) at the end of the EC prevents new asynchronous transmissions to be started, to avoid EC overruns.

C. DIFFERENCES OF THE BASE PROTOCOLS

Independently of the numerous differences concerning traffic handling mechanisms, both protocols share sufficient resemblances to establish a baseline for the comparison we aim at. Both are based on standard Ethernet frames and both have dynamic traffic segregation and bandwidth reservation mechanisms that allow guaranteeing the quality of service (QoS) requirements of real-time message streams.

IV. ADMISSION CONTROL PROTOCOLS

The AC process consists of evaluating whether the resources available in the network are enough to transmit a given set of streams within their specified QoS parameters. In this section, we briefly describe how each protocol triggers the AC and manages the propagation of the respective decisions through the network.

A. AC PROTOCOL OF AUDIO VIDEO BRIDGING

The AC in AVB is carried out via SRP, following a distributed approach. FIGURE 5 sketches the operation of the protocol in a line topology with one talker and one listener connected through two bridges. The talker is responsible for announcing its intention to transmit and to trigger the resource reservation process through a special message called *talker advertise* (T_advertise in FIGURE 5). The talker advertise message conveys information about the resources needed by the talker to communicate, *i.e.*, it conveys the stream's parameters.

Once a bridge receives a talker advertise message, it checks the schedulability of the stream, *i.e.*, it executes the admission control test. If the stream can be scheduled, the talker advertise is forwarded to the following bridges until reaching the end-stations.

Whenever a successful talker advertise reaches an end-station willing to become a listener of the stream, the listener issues a *listener ready* message (L_ready in FIGURE 5). When a bridge receives a listener ready message, it checks the availability of resources again. If resources are still available, the bridge locks the resources and forwards the listener ready message to the port through which the talker advertise

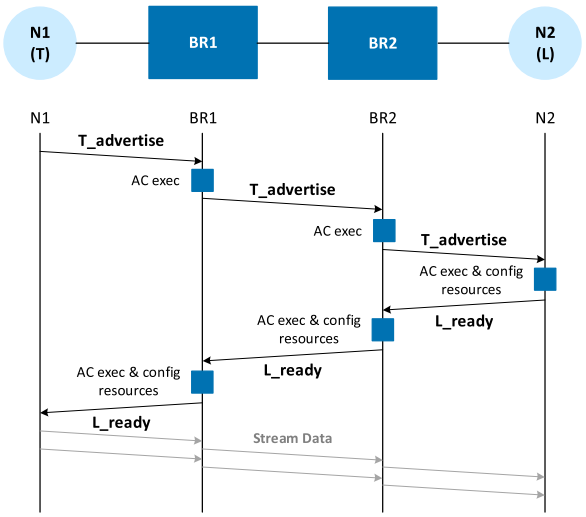

FIGURE 5. Operation of SRP admission control process.

was received. Finally, if the reservations are successful across the path between the talker and a listener, the talker receives the listener ready message and the stream is established. This way, the talker can start transmitting frames through the established path.

The creation of the stream may fail due to the lack of resources in the path between the talker and listener(s). In such case different messages are issued (*talker advertise failed*, *listener asking failed* and *listener ready failed*) to signal the origin and type of failure (bridges have insufficient resources, no listener has sufficient resources and some listeners have insufficient resources, respectively). Further details in this regard can be found in [24].

The talker advertise and listener ready messages are part of the control plane of SRP. These messages are transmitted through a control channel, which guarantees a minimum bandwidth availability to manage streams. Moreover, SRP also provides a means to limit the number of requests that nodes can transmit within a time interval. This prevents bursts that could jeopardize the transmission of data and other control messages. Specifically, SRP guarantees that any request will be transmitted within 200ms from its arrival to the output queue, while each device cannot transmit more than three request messages within 300ms. SRP allows encapsulating several requests in a single frame to reduce the impact of the limitation on the number of request messages and reduce the overhead.

B. AC PROTOCOL OF HaRTES

HaRTES employs a producer-consumer model, requiring that both producer and consumer nodes issue resource reservation requests to instantiate a message stream. These requests, stating the asynchronous streams' properties, are analyzed by the admission control unit (ACU) inside the master, which applies a schedulability analysis to decide upon acceptance. If the request can be satisfied, *i.e.*, there are enough resources to serve the stream without compromising the timeliness

of existing reservations, the switch allocates the required internal resources, *e.g.*, memory and servers. These resources are then automatically configured according to the declared attributes of the associated streams, *e.g.*, maximum message size, minimum inter-arrival time, and deadline.

For multi-hop networks, the schedulability analysis, briefly introduced in Section V-B, requires global knowledge of the network topology, *i.e.*, the set of links connecting nodes to switches and switches to switches (interlinks), as well as system-wide message streams. Therefore, the AC is centralized in one particular master, the *ACMaster*, selected among all masters through a system-wide configuration parameter. This implies that reservation-related messages received by any local master are forwarded to the ACMaster, which thus has global knowledge of the network and existing stream reservations.

The whole process is shown in FIGURE 6. Producer/consumer nodes send a request (*Add_Msg*) to their local master within the AW. Requests received by regular masters are relayed to the ACMaster. Upon receiving a request from the producer and the consumer of a given stream, the ACMaster runs the AC tests. If the test fails, the reservation is declined and all nodes are informed. Otherwise, the ACMaster sends commands (*Config_Msg*) to all switches along the stream's path to configure the necessary switch resources. Finally, upon receiving a positive reply from all the aforementioned switches, the ACMaster notifies (*Ack*) the producer/consumer nodes, which can then start exchanging data messages.

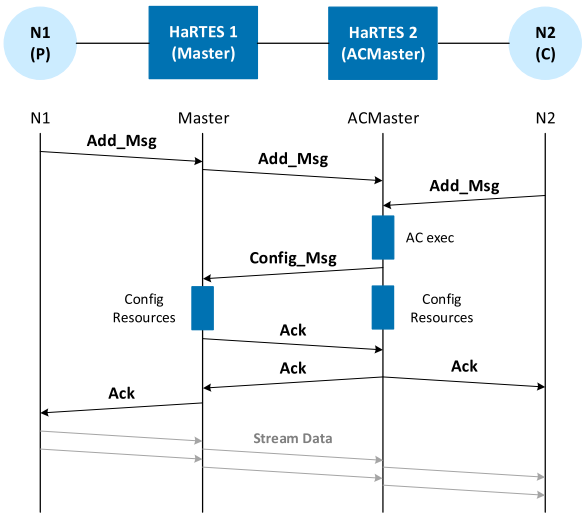

FIGURE 6. Operation of HaRTES's admission control process.

All message exchanges related to the aforementioned reservation process are performed via a default (asynchronous) virtual control channel that is created at boot time. To control burstiness while ensuring a desired reaction time to stream (re)configuration requests, this channel is associated with a suitable capacity and priority, to meet the application needs (*e.g.*, assigning it with the highest priority maximizes reactivity at expenses of higher interference on application traffic).

C. DIFFERENCES IN THE AC PROTOCOLS

The nature of the base protocols strongly influences how the AC is carried out by each one. AVB follows a distributed model, where the talker is responsible for triggering the creation of the stream; listeners respond to the talker, and all the components in the stream's path carry out the AC. Conversely, HaRTES counts with a central component to make the decisions, including the AC test. Thus, the producer and the consumer(s) of a stream are responsible for announcing their intention to communicate and the network switches relay this information to the ACMaster that takes the acceptance decisions.

This leads to significant differences between the protocols. Specifically, HaRTES requires a complete view of the network, whereas AVB does not. On the other hand, HaRTES executes the AC only once, regardless of the number of switches in the network, whereas AVB must execute the AC twice for each switch. Moreover, HaRTES allows to tune the control channel according to the application needs, namely by the customization of its priority and bandwidth, while in AVB the control channel attributes are fixed. As it will be seen, these differences have a significant impact on the reactivity of the protocols and on the influence that operating conditions (*e.g.*, network load, number of streams, packet size) have on it.

V. ADMISSION CONTROL TESTS

To assure continued real-time behaviour, the admission control evaluates each change request to verify if the available network resources are sufficient to satisfy the requirements of the resulting message set. This procedure is commonly known as a schedulability test and can be responsible for a significant part of the AC process latency. For this reason, and for the sake of completeness, this section briefly overviews the tests carried out by the two protocols. Note that the efficiency of these tests in terms of bandwidth utilization is out of the scope of this paper. The purpose of this section is solely to allow understanding their contribution to the AC process latency.

A. AC TEST OF AUDIO VIDEO BRIDGING

As previously mentioned, in AVB the AC is fully distributed, so all bridges in the path between a talker and listener(s), as well as the end-systems, check the availability of resources locally and then inform the other devices on the local result. Stream attributes are defined in Equation 3.

$$S_i \equiv \{Id_i, mac_i, vlan_i, Msize_i, Mint_i, P_i, R_i, Mlat_i\} \quad (1)$$

Id_i is the unique identifier of the stream; mac_i is the multi-cast destination MAC address; $vlan_i$ is the Virtual LAN of the AVB domain through which data messages are transmitted; $Msize_i$ is the maximum size of the messages to be transmitted through the stream; $Mint_i$ is the maximum number of frames that the talker can transmit during the *class measurement interval* assigned to the stream class; P_i is the priority of the traffic class; R_i indicates if the stream

belongs to the *emergency* type, which has higher priority in the reservation than the rest of streams of its class; and $Mlat_i$ conveys the maximum accumulated latency of the stream as it traverses the network. The information that is not conveyed by the talker advertise is calculated by the bridge, *e.g.*, the time required by the bridge to transmit a frame; or available as configuration parameters, *e.g.*, the transmission interval of a class. AVB defines two stream reservation (SR) classes: SR class A and SR class B, with class measurement intervals predefined as 125 and 250 μs, respectively [25].

The AC analysis is carried out separately in each bridge. When a bridge receives a talker advertise it increases $Mlat_i$ by adding the time required to transmit the largest frame of the stream, denoted by M_i. Specifically, the bridge accounts for all interfering frames; the largest blocking frame; the worst-case time required to forward frame M_i from the input to the output port; the propagation time of the link and the time required by the media to become available for transmission. The talker advertise is then forwarded with the new $Mlat_i$ value and the process is repeated in each bridge.

Listeners use $Mlat_i$ to evaluate whether the latency of the stream is too large, rejecting the request if that is the case. The value calculated for $Mlat_i$ cannot be modified during system operation. If the application wants to modify it, the stream must firstly be withdrawn from the network and then created again, with the new attributes in place.

B. AC TEST OF HaRTES

In HaRTES networks, the traffic requirements are expressed by the set Γ of N_s real-time streams (Equation 2), which follows the traditional periodic/sporadic model [30]. In this model, each stream S_i is defined by its transmission period T_i (minimum inter-arrival time for event-triggered traffic), transmission time C_i of its maximum sized frame, priority P_i, initial offset O_i (ignored for event-triggered traffic), and relative deadline D_i. L_i is the set of links crossed by stream S_i. These parameters are either directly conveyed by producer/consumer reservation requests, *e.g.* P_i and D_i, or derived from existing information, *e.g.* L_i from the network topology and the IDs of producers and consumers and C_i considering the maximum frame size Λ_i, the links speed and all overheads including start-of-frame (SOF), frame check sequence (FCS) and the inter-packet gap (IPG).

$$\Gamma = \{S_i | S_i = (T_i, \Lambda_i, C_i, P_i, O_i, D_i, L_i), i = 1, 2, .., N_s\} \quad (2)$$

Upon receiving a stream request from the producer and at least one consumer node, the ACMaster first checks the request correctness, *e.g.*, number and consistency of attributes, immediately rejecting invalid requests. Then, it computes the end-to-end worst-case response time (WCRT), *i.e.*, the time lapse between the instant when a frame becomes ready at the sender interface and the latest instant when its reception at the receiver interface terminates, for the new stream. If the estimated WCRT is higher than the stream deadline, the request is rejected, otherwise, the WCRT for every other stream currently in the network is computed to account for possible interference due to the new stream. If the deadline requirement of any given stream is found unmet, the ACMaster promptly stops the AC process and rejects the request, else, the new stream is accepted and integrated in the global traffic requirements.

To compute the WCRT for a given stream, the ACMaster employs the analysis presented in [28] for HaRTES multi-hop networks in reduced buffering scheme (RBS) [28] mode. In summary, the computation employs the classical response time analysis based on the accumulation of delays within iterations (Equation 3) to estimate the response time of frames as they are transmitted through links, from the producer to the consumer node.

$$rt_{i,a,b}(x) = \frac{C_i}{\alpha_{i,a,b}} + I_{i,a,b} + B_{i,a,b} + SD_{i,a,b} \quad (3)$$

The calculation considers the transmission time C_i of the S_i frame itself, as well as three types of interference that may arise from the existence of other streams sharing common links: (i) $I_{i,a,b}$, the interference from messages with higher or equal priority; (ii) $B_{i,a,b}$, the blocking from messages with lower priority; (iii) $SD_{i,a,b}$, the total switching delay for the stream across the entire route. An inflation factor $\alpha_{i,a,b}$ adjusts the transmission time of frames according to the respective confinement window (synchronous for time-triggered, and asynchronous window for event-triggered streams) [31]. The three interference components depend non-linearly on the interval of time that is being considered (x). Thus, to solve the non-linear Equation 3, a fixed-point iterative process is used. The response time is obtained when the iteration process converges, *i.e.*, $rt_{i,a,b}(x) = rt_{i,a,b}(x-1)$, with the first iteration computed as $rt_{i,a,b}(0) = \frac{C_i}{\alpha_{i,a,b}}$. The iterative process terminates either in case of convergence or if $rt_{i,a,b}(x) > D_i$ (deadline violation). Further details can be found in [28].

C. DIFFERENCES OF THE AC TESTS

AVB is class-oriented, relying on two traffic classes to provide different timing guarantees. Thus, all streams that belong to one class are treated in the same way and are provided with the same type of guarantees. On the other hand, HaRTES is stream-oriented, that is, different streams may have different guarantees, as defined *e.g.*, by their priority and server capacity. Moreover, HaRTES takes into account all existing streams when carrying the AC of a new request. If the creation of the new stream can jeopardize the timeliness of existing ones, the stream is not created. Conversely, AVB does not take this into consideration, which could lead to the violation of the guarantees of existing traffic. Finally, the AC test of HaRTES is holistic while the one of AVB is based on local information. As such, HaRTES analysis is potentially more accurate, despite less scalable.

TABLE 1 summarizes the main differences between HaRTES and AVB regarding both the schedulability tests, discussed in this section, as well as architecture and AC protocols, presented in the two previous sections. As it can

TABLE 1. Main qualitative differences between HaRTES and AVB.

Protocol	Base protocol	AC protocol			AC test		
	Architecture	Network view	Control channel	Test execution	Test	Traffic evaluated	Traffic classification
HaRTES	Centralized	Complete	Custom	Once	Holistic	All streams	Stream-oriented
AVB	Distributed	Local	Predefined	2xSwitch	Local	Requested stream	Class- oriented

be seen, the protocols differ significantly in all considered aspects, indicating that the performance of both protocols should reveal important differences.

VI. EXPERIMENTAL SETUP AND RESULTS

In this section we evaluate, experimentally, the latency of executing reconfiguration requests in a distributed (AVB) and in a centralized (HaRTES) admission control architecture. We seek to understand the origin of latency components and the latency dependency with network/system parameters and reconfiguration scenarios.

A. EXPERIMENTAL SETUP

The experimental setup used in this paper closely follows that used by *Meyer et al.* [16], with a few minor differences. Namely, the number of nodes is reduced from 10 to 5, due to a reduced number of Ethernet ports on the available HaRTES switches. Consequently, the streams' source and sink nodes were also adapted. Nevertheless, the network topology and total load were kept similar.

The network topology used in the experiments is depicted in FIGURE 7, comprising five end nodes (N1-N5) and three switches (Sw1-Sw3) interconnected in a line topology. The switches (aka bridges) are either HaRTES or AVB. The ACMaster (ACM) is present in the HaRTES network, only, and runs on an i7-4770 CPU with 8GB DDR3 RAM and Arch Linux 4.20.7 OS, being connected to a port of HaRTES Sw2. The HaRTES switches are built on commercially available NetFPGA boards featured with 4 Ethernet ports. AVB bridges are commercially available equipment based on the 400MHz ARM9 Marvell 88E7251 SoC, developed by dsp4YOU and certified by the AVNU Alliance, while end nodes rely on Intel i210-T1 network cards which implement all AVB standards. All links operate at 100 Mbit/s in both networks. Moreover, we use two hardware sniffers (Hilscher netANALYZER NANL-C500-RE and NANL-B500G-RE), both with nanosecond resolution, to concurrently capture and timestamp frames in multiple links.

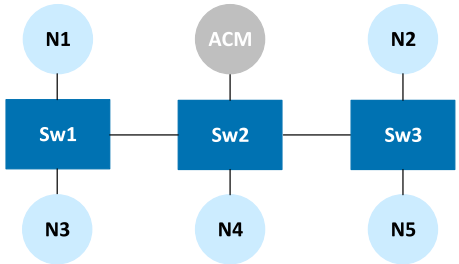

FIGURE 7. Topology of the experimental setup.

We claim this setup is significant for our purposes of comparing the two opposite architectural paradigms of AC protocols in the scope of reconfigurable real-time Ethernet, either applied to vehicles [3], [13], [32]–[34], autonomous robots [1], or factory cells [4]. Our setup has a lower number of nodes than typical, but, for our purposes, the number of nodes is not a significant parameter, since it has negligible influence in either of the protocols (see Sections IV and V). On the other hand, the number of switches (network depth in particular), the network load, and the number of streams play a significant role in the AC architectures we are considering. Concerning the number of switches, according to the literature referred above, many real systems in the said domains have 3 or fewer switches. Thus, we consider our setup to be realistic. Concerning network load, we use broadcast traffic to cover a broad configuration space (going above 90%). Finally, we use a relatively small number of streams. This is a limitation arising from the number of reservations that the AVB accepts in this setup. Despite low, the results achieved with such a number of streams already provide relevant insights on the impact of the AC architecture. Moreover, remember that each of these streams typically carries an aggregation of multiple signals [34]. Nevertheless, we consider streams with a relatively small payload, which is typical for control traffic. This is also in line with the Maximum Transmission Unit (MTU) of 256B that is recommended in the TSN standard.

B. AC LATENCY: VARIATION WITH OPERATIONAL RECONFIGURATION SCENARIOS

In this section, we aim at assessing the temporal behavior of the AC protocols in reconfiguration scenarios that are common in practice, namely issuing bulk reservations at startup, issuing sequential reservations at runtime, and issuing bulk reservations at runtime, too. These scenarios are detailed below, considering the concrete streams used.

TABLE 2 shows the parameters for the traffic used in these experiments, where T is the frame period and P the frame payload. Flows F1 and F8 represent best-effort (*BE*) background broadcast traffic, while flows F2 to F7 are the time-sensitive data streams produced, for example, by different kinds of sensors that belong either to *Class A* (higher priority) or *Class B* (lower priority). To maximize the interference and number of links crossed, the flow transmitters are all placed on one side of the network (Sw1) whereas the receivers are located on the opposite side (Sw3).

The transmission periodicity of streams F2 to F7 is restricted by the class measurement interval of AVB's SR classes (Section IV-A). Therefore, streams with the highest

TABLE 2. Traffic characteristics for the reconfiguration scenarios.

ID	Source	Sink	Class	T (μs)	P (bytes)
F1	N1	Broadcast	BE	250	256
F2	N1	N2	Async/Class B	250	64
F3	N3	N2	Async/Class B	250	64
F4	N3	N2	Async/Class A	125	32
F5	N1	N5	Async/Class A	125	32
F6	N1	N2	Async/Class A	125	32
F7	N1	N2	Async/Class A	125	32
F8	N4	Broadcast	BE	250	256

rate are assigned to SR class A ($T = 125\mu s$) while class B is set for streams with lower rate requirements ($T = 250\mu s$). HaRTES poses no constraints on the periodicity of traffic other than being a multiple of the elementary cycle (EC) length. Therefore, for the sake of comparison, the EC on all HaRTES switches is configured with length equal to the minimum period value found in the stream set (125 μs).

Additionally, since AVB does not provide explicit support to time-triggered traffic, in HaRTES the length of the synchronous window is set to $0\mu s$ and all RT streams are assigned to asynchronous servers. Each server is configured with T_s as the associated stream's period and C_s as the stream's payload transmission time including all overheads from the PHY up. To mimic the prioritization of AVB ($Priority\{classA\} > Priority\{classB\}$), servers associated with streams F4 to F7 are set with a higher priority level than those serving streams F2 and F3. Finally, a server stream (*AC Server*) with a T_s of $125\mu s$, C_s of 200 bytes and the highest priority level, is assigned to AC control transactions. The TM and turn around windows are both set to $10\mu s$. We also consider the AC requests are sent approx. at the same time by the producer and consumer so that, when the AC result notification arrives at the producer it can start transmitting immediately (FIGURE 6).

Considering this traffic, we now detail the three experimental scenarios we will use, as follows:

- *Scenario I* - Nodes simultaneously request the admission of the six RT streams at startup, representing the initialization of a complex application;
- *Scenario II* - Nodes send requests for RT streams sequentially (F2→F5→F6→F7→F3→F4) with 1 second of delay between each request and starting immediately, representing simple services that are sequentially activated;
- *Scenario III* - Nodes trigger two groups of simultaneously-triggered stream requests: G1 = {F5, F3, F4} and G2 = {F2, F6, F7}. G1 is launched first and starting immediately, then G2 10 seconds after, emulating the activation of complex services at runtime.

For each scenario, we measured the AC latency, *i.e.,* the time elapsed between the transmission of a stream admission request (talker advertise message in AVB, producer request message in HaRTES) and the reception of the respective acknowledgment message (listener ready message in AVB, acknowledgment message from the ACMaster in HaRTES) at each stream's producer node. The results of the AC latency obtained after repeating each scenario 1000 times are shown in TABLE 3. Note that the BE flows (F1 and F8) are being transmitted during the duration of the whole experiment, creating additional interfering traffic.

TABLE 3. Admission control latency in the three reconfiguration scenarios.

Protocol	Stream ID	Response time (ms) Min.	Mean	Max.	Std. dev. σ
Scenario I - Simultaneous Requests					
HaRTES	F2	29.10	38.52	62.00	9.04
	F3	29.17	38.55	62.08	9.02
	F4	31.17	38.70	61.17	9.04
	F5	31.24	38.69	62.13	9.03
	F6	31.28	38.79	62.18	9.06
	F7	31.39	38.84	62.28	9.04
AVB	F2	220.76	501.61	883.89	96.39
	F3	260.91	495.43	923.29	95.69
	F4	260.89	495.11	923.27	95.87
	F5	220.75	527.36	922.96	116.06
	F6	220.63	501.58	864.19	96.49
	F7	220.77	501.18	864.09	96.92
Scenario II - Sequential Requests					
HaRTES	F2	4.97	6.62	9.67	1.64
	F5	5.55	7.17	10.79	1.85
	F6	5.95	7.77	11.67	2.01
	F7	6.17	8.14	12.62	2.12
	F3	6.79	9.09	13.79	2.43
	F4	7.05	9.34	14.47	2.55
AVB	F2	160.54	264.07	982.81	68.93
	F5	140.45	247.38	561.89	48.64
	F6	160.49	249.88	822.79	48.42
	F7	240.62	259.19	762.58	64.67
	F3	140.52	268.49	782.85	77.08
	F4	160.57	257.53	561.95	57.84
Scenario III - Grouped Requests					
HaRTES	F5 (G1)	15.21	21.83	34.10	5.72
	F3 (G1)	15.28	21.91	34.17	5.73
	F4 (G1)	16.42	21.98	34.24	5.74
	F2 (G2)	16.84	23.07	36.25	6.11
	F6 (G2)	17.29	23.13	36.34	6.11
	F7 (G2)	18.42	23.20	36.91	6.13
AVB	F5 (G1)	140.49	275.58	662.28	81.93
	F3 (G1)	220.76	309.91	822.86	85.57
	F4 (G1)	200.90	309.48	802.84	85.24
	F2 (G2)	160.58	295.38	782.06	38.60
	F6 (G2)	140.50	295.09	782.05	38.98
	F7 (G2)	140.49	294.79	782.77	38.58

Values obtained from 1000 samples per experiment condition.

Starting with Scenario I we note that each architecture exhibits approximately the same latency when creating all the reservations. This is expected since the requests are not issued with tight synchronization and interleaving may occur along the path without enforcing order consistency. Thus, in each run of the experiment, the order in which the requests are processed varies. Concerning the maximum observed values, which are especially relevant to characterize the AC real-time behavior, HaRTES shows around 63% increment over the average, while AVB shows around 80%. These values correspond to when a request arrives last and has to wait for the processing of all other requests. Thus, it suffers a significant queuing delay beyond the processing of the request itself.

Another aspect related to the different architectures is the variability of the measured values. The centralized AC case (HaRTES) has significantly less variability affecting

the maximum observed values (approx. 1.8%) than the distributed one (approx. 6.5%). The same happens with the span of observed values. This seems to confirm that the distributed AC (AVB) is more susceptible to interference than the centralized one, which is expected given the more complex protocol, involving more control messages and more tests. Moreover, another contributing factor is the separate high priority configuration channel used by HaRTES that does not exist in AVB.

Finally, just a quick note on the absolute AC latency values, which are more than one order of magnitude larger in AVB than in HaRTES. This magnitude is explained both by architectural and technological parts; we will further address these aspects in the next section.

The results obtained for Scenario II allow us to eliminate the queuing delay affecting the AC requests in Scenario I. In this case, each request is handled separately as soon as it arrives. Note that the separation between consecutive requests is larger than the maximum observed AC latency. Naturally, both architectures now respond faster, but the difference is much larger in the centralized AC than in the distributed one, with a strong reduction in both average and maximum observed values. The centralized AC (HaRTES) also shows a clear increase of the AC latency with the number of installed streams, as expected, due to the longer time taken by the AC test. Conversely, the distributed case shows a wider variation with just a marginal correlation with the increase in installed reservations in the later requests. This reduced sensitivity is expected since the AC test in AVB is class and bandwidth-oriented, thus depends primarily on the number of crossed switches, which is constant in this experiment. We will see, later on in TABLE 5 (Exp.III AVB[T-L]), that this latency is dominated by the AC process in the target end node. Curiously, the average results also seem to indicate a marginal impact of the stream class.

Finally, the results regarding Scenario III show an intermediate case with some level of queuing delay affecting the AC latency. The results are compatible with Scenarios I and II. The centralized AC (HaRTES) shows rather steady values for average and maximum observations within each group. Moreover, the second group shows an extra AC latency, due to the additional reservations already in place. Again, the distributed AC (AVB) shows wide variability that cannot be correlated to the increase in the installed reservations.

C. AC LATENCY: VARIATION WITH NETWORK AND SYSTEM PARAMETERS

In order to better understand how the AC latency varies in each architecture, we carried out three experiments as a short sensitivity study on the impact of three key factors:

- *Experiment I* - network size;
- *Experiment II* - network load;
- *Experiment III* - number of streams.

To isolate the effect of each one of these factors on the AC latency, we adopted customized traffic sets. To assess the impact of the network size, we measured the time taken to create stream F7 (TABLE 2) modified with the listener/consumer placed in different nodes generating growing hop counts (N1→N3: 2 hops, N1→N4: 3 hops, N1→N5: 4 hops). There was no other traffic present. To evaluate the impact of network load, we measured the time that it takes to create stream F7 (N1→N2, involving 4 hops) with a network load of 0%, 50% and 90%. The interfering traffic is generated by Node N3 and transmitted in broadcast using the best effort class. Furthermore, we considered two different interfering payload sizes (256 and 1000 bytes) with periodic arrival to evaluate different blocking and processing times. For each payload, we adjusted the transmission period according to the target load. TABLE 4 summarizes the traffic characteristics for these experiments.

TABLE 4. Message set for evaluating the impact of network size and load on the creation of stream F7.

Experiment	Stream F7			Interference	
	Source	Sink	Class	Load	P (bytes)
Network size	N1	N3	Async/A	0%	N.A.
	N1	N4	Async/A		
	N1	N5	Async/A		
Network load	N1	N2	Async/A	0%	N.A.
				50%	256
					1000
				90%	256
					1000

Then, we also evaluated the impact of the number of pre-installed streams on the AC test. To that end, for HaRTES, we measured the time elapsed between the reception of a request at the ACMaster and the output of the AC analysis. For AVB, we measured the forwarding delay (AVB [T] in TABLE 5) of Talker Advertise (TA) messages in a single bridge (Sw1), which corresponds to the AC test execution in that bridge. Additionally, we also measured the forwarding delay of Listener Ready (LR) messages (AVB [L]), which corresponds to the AC test plus resources reservation in that bridge. Then, we measured the time interval between the transmission of a TA and the reception of the respective LR message in the same bridge (Sw1), corresponding to the nested execution of tests and resources reservation along the path. The purpose of these last two measurements will be clear over the course of the following discussion. Concerning the pre-installed streams, we replicated streams F4 and F5 (TABLE 2) as needed until we reached the maximum number of streams that AVB was able to support in the current configuration (7 pre-installed plus stream F7).

TABLE 5 summarizes the results obtained with samples of 2000 individual measurements. We also present the Empirical Cumulative Distribution Function (ECDF) for the AC latency observed in Experiments I, II, and III in FIGURES 8-10, respectively. Their vertical axis represent an estimation of the probability (P(X<=x)) of the latency in the horizontal axis being lower than a certain value x. Thus, the ECDFs give us a complete view of the distribution of the measurements.

TABLE 5. Admission control latency for Experiments I, II, and III.

Protocol	Experiment Conditions	Response time (ms) Min.	Mean	Max.	Std. dev.σ
Experiment I - Number of Hops					
HaRTES	2 Hops (F7:N1→N3)	4.19	5.99	10.43	0.91
	3 Hops (F7:N1→N4)	4.61	6.50	11.74	1.08
	4 Hops (F7:N1→N5)	4.96	6.91	12.18	1.16
AVB	2 Hops (F7:N1→N3)	120.42	217.34	221.14	15.28
	3 Hops (F7:N1→N4)	140.48	240.91	521.78	19.65
	4 Hops (F7:N1→N5)	140.48	248.75	782.64	49.28
Experiment II - Network Load					
HaRTES	No Load	4.96	6.91	12.18	1.16
	256 PL & 50% NL	4.90	6.95	11.52	1.19
	256 PL & 90% NL	4.89	7.16	10.52	1.27
	1000 PL & 50% NL	4.87	6.83	12.25	1.13
	1000 PL & 90% NL	4.95	7.02	12.25	1.20
AVB	No Load	140.48	248.75	782.64	49.28
	256 PL & 50%NL	180.85	291.25	682.34	49.68
	256 PL & 90%NL	180.63	326.96	943.32	50.69
	1000 PL & 50%NL	180.53	267.89	722.53	39.86
	1000 PL & 90%NL	180.52	278.94	782.73	63.43
Experiment III - Number of Existing Streams					
HaRTES	1 Stream	4.58	6.00	9.67	0.98
	3 Streams	5.03	6.30	10.08	1.76
	5 Streams	5.36	6.98	10.69	1.96
	7 Streams	8.58	10.64	17.40	3.04
AVB [T]	1 Stream	2.26	2.54	3.28	0.17
	3 Streams	2.34	2.58	3.49	0.19
	5 Streams	2.32	2.59	3.36	0.18
	7 Streams	2.38	2.62	3.44	0.17
AVB [L]	1 Stream	6.24	7.11	8.56	0.41
	3 Streams	6.07	7.04	8.74	0.40
	5 Streams	6.09	7.08	9.03	0.40
	7 Streams	6.18	7.16	8.20	0.41
AVB [T-L]	1 Stream	13.84	114.25	411.12	35.82
	3 Streams	25.96	122.57	415.97	28.36
	5 Streams	18.54	111.59	415.96	24.96
	7 Streams	18.22	112.50	417.38	27.46

Values obtained from 2000 samples per experiment condition.

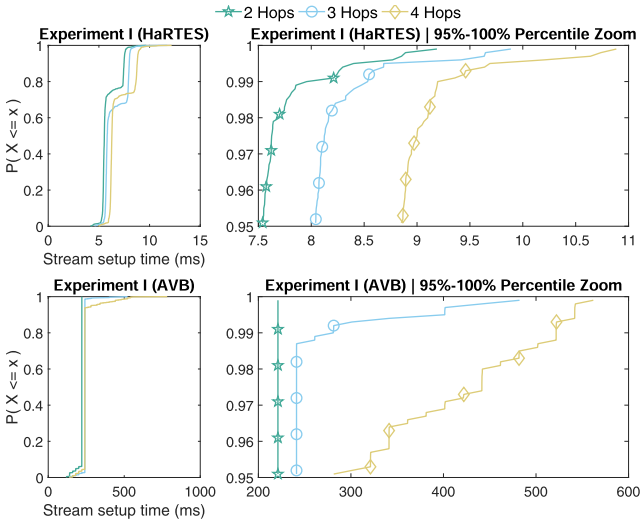

FIGURE 8. ECDF (complete - left, 95 percentile and above - right) of the admission control latency for Experiment I, impact of network depth.

Regarding Experiment I, we observe that the AC latency increases with the number of hops as expected, due to the growing number of switches that have to be configured (1, 2 and 3, respectively). This is particularly relevant in the distributed AC and especially visible in the maximum observed values. The corresponding EDCF (FIGURE 8) shows a strong concentration around approx. 200ms, with a relatively long tail. The centralized AC is affected, in addition, by the AC test, whose complexity also depends on the number of hops that streams have to cross [28]. However, as seen before, the total AC latency of the centralized AC is significantly lower than with the distributed AC. The ECDF in FIGURE 8 shows a strong concentration around two values, approx. 6ms and 8ms. This variation arises from the execution time of the AC test in the ACMaster. Finally, the values observed in this experiment are also compatible with the observations in Scenario II reported in the previous section.

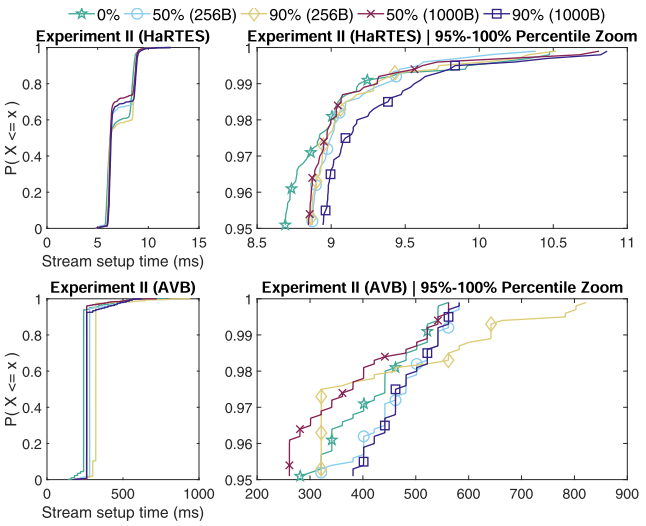

FIGURE 9. ECDF (complete - left, 95 percentile and above - right) of the admission control latency for Experiment II, impact of best-effort network load.

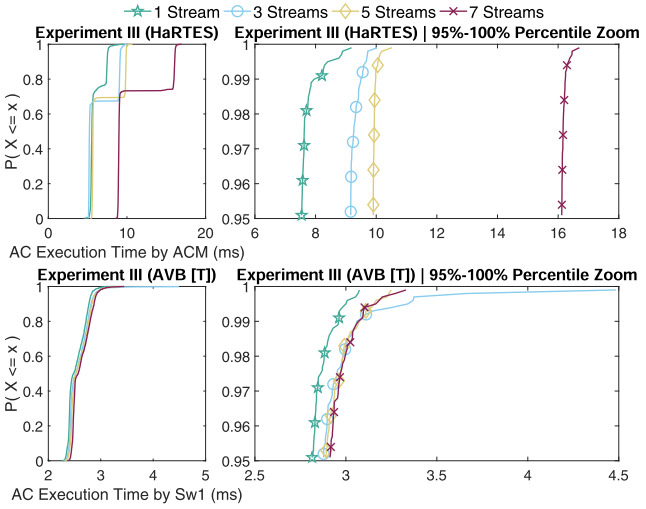

FIGURE 10. ECDF (complete - left, 95 percentile and above - right) of the admission control latency for Experiment III, impact of existing streams.

The results of Experiment II show that the AC latency is affected by network traffic essentially in the distributed AC. The impact on the centralized AC is residual. Note in FIGURE 9 how the diverse curves overlap with the reference case (no load) in the vertical parts. In fact, the AC messages in HaRTES use a protected high priority channel,

thus the network load only impacts the AC process through packet blocking. This is visible in the maximum observed values, which occur with the largest payload, independently of the network load value. Concerning the distributed AC, the ECDF of this experiment shows a sharp concentration on values that are around 250ms. The curves corresponding to network load interference using shorter packets are shifted to the right, particularly that of 90% load, showing higher impact. This suggests that the impact results from packet processing overhead at bridges and end nodes.

Experiment III shows the execution time of the AC test. In the centralized AC (HaRTES), the similarity of these results with those of the previous experiments shows that the AC latency is clearly dominated by the execution time of the AC test. Moreover, the specific schedulability analysis [28] (see Section V-B) has a pseudo-polynomial complexity on the number of streams. Thus, we see a significant increase when the reservations set increases and particularly with 7 streams. In the distributed AC case (AVB), the three components we measured allow us assessing the AC test execution time in each bridge (AVB [T]) as well as the time to test, allocate and reserve resources (AVB [L]). We observed that these values, in bridge Sw1, are relatively short and steady, 3.5*ms* and 9*ms* respectively in the maximum case, without visible impact of the number of streams already in place. This is expected since these operations are independent of the number of streams (Section V-A). Then, we also measured the round trip time of the reservation process from bridge Sw1 to the Listener and back (AVB [T-L]). The observed values increased significantly, suggesting that the AC test and resources allocation and setup at the end node take the largest part of the total AC latency of the distributed AC case and are responsible for its large variation.

D. COMMENTS ON THE RESULTS

At this point, it is important to recall that the purpose of this work is the comparison of AC architectures, not just a comparison of AVB versus HaRTES. Therefore, the magnitude of the observed values is just an indication, tied to the specific choices made in the setup, but more importantly we looked at the variation and decomposition of the AC latency. Here we express our main takeaways.

Starting with the distributed AC architecture, here represented by AVB, it is composed of multiple local AC tests, executed twice at each bridge in the sender-receiver path. Due to these multiple steps, the potential to suffer interference, both computing and communication, is higher than with a centralized AC architecture in which the requests are directly forwarded to the ACMaster. This leads to high variability of the AC latency.

Moreover, because of the potentially high number of times the AC test is executed in the distributed AC, it might be impractical to use more accurate tests, *e.g.*, response-time based, that have pseudo-polynomial complexity. For this reason, distributed AC typically uses aggregated tests, *e.g.*, bandwidth-based, which have constant complexity. This is the case of AVB, and for this reason, we did not observe a variation of the AC latency with the number of installed streams, which is a plus for scalability.

Probably, the most relevant observation we made concerning the AC latency with distributed AC architecture is its dependence on the capacity of the end nodes to execute the AC test plus resource allocation and setup. This constraint applies to all end nodes and it is thus, a strong constraint. The AVB node interfaces we used have a clear poor performance that can easily jeopardize the real-time behavior of the AC process. In our setup, if the AVB node interfaces had an AC performance similar to that of the AVB bridges, we could have achieved maximum AC latency of the order of 50ms, which would still be larger but already comparable with the AC latency achieved with the centralized AC architecture. Unfortunately, we could not find any information on the AC performance of other alternative AVB node interfaces.

The centralized AC architecture was represented by HaRTES. The single AC test allows using more accurate analysis, such as per-stream response time-based. This kind of test, on the other hand, can cause scalability problems, which, in turn, can be mitigated using a powerful processor to carry out this function. In HaRTES, this option allowed obtaining comparatively low AC latencies and with relatively low variability, which is very important for the AC real-time behavior. As a quick assessment of the scalability of the AC test in HaRTES, we can refer to the work in [35], in which we report an AC test execution time below 170ms for 100 streams in a network of 16 switches.

There is an ample choice of AC tests, allowing different trade-offs between accuracy and execution time, that can be selected to handle more complex scenarios were scalability becomes an issue. Moreover, if reliability is a concern, more AC masters can be added to the system in an adequate replication scheme.

One final aspect that our comparison allowed observing is the benefit of using a protected high-priority channel to convey the AC transaction. We believe this is mandatory for isolating the AC process from the application network load, contributing to a deterministic AC latency and a strong real-time behavior of the AC protocol.

VII. CONCLUSION

Emerging applications, such as Industry 4.0 and Autonomous Driving, have to be flexible, adaptive and resource-efficient, while still assuring real-time behavior and determinism. This requires an admission control in place, to filter out reconfiguration requests that can not be served with the required QoS. The AC process, though, also needs to be real-time, and its timing behavior depends strongly on its architecture. Therefore, in this paper we compared the AC processes of two real-time reconfigurable Ethernet protocols that use opposite AC architectural paradigms. Namely, we use AVB with a distributed AC and HaRTES with a centralized AC.

We experimentally evaluated the reactivity of these protocols facing reconfiguration requests that trigger the respective

AC processes in a small but representative setup. The results obtained show that the centralized AC (HaRTES) consistently outperformed the distributed one (AVB) in latency and jitter of the admission control processes by more than one order of magnitude.

More importantly, we realized the participation of the end nodes in the distributed AC can be very detrimental of the stability and real-time behavior of the AC process. This requires a careful choice of the nodes network interfaces.

We also observed the positive impact on the AC latency of having an isolated high-priority channel to convey the AC transactions. As this introduces a negative impact on the performance of real-time traffic sharing the same network path and thus, on the overall system schedulability, its design should be tailored to the target application needs (schedulability vs AC responsiveness trade-off). In this regard, the ability of the distributed approach to exploit disjoint paths for the AC process may make it more efficient in very specific network topologies and system configurations, as the centralized approach concentrates all AC flows into a single node.

Overall, our results corroborate the importance of the fully centralized and centralized network/ distributed user resource reservation models that TSN (AVB's successor) is now considering. We believe that these models can bring the improvements needed for TSN to cope with stringent reconfiguration requirements. When the necessary hardware and configuration tools become available, we plan to extend this study to include TSN's new dynamic reservation models.

REFERENCES

[1] (Jan. 2020). *Real-Time Adaptive Networked Control of Rescue Robots*. [Online]. Available: https://cordis.europa.eu/project/id/318902

[2] (Jan. 2020). *SAE Levels of Autonomy*. [Online]. Available: https://www.automotivelectronics.com/sae-levels-cars/

[3] S. Samii and H. Zinner, "Level 5 by layer 2: Time-sensitive networking for autonomous vehicles," *IEEE Commun. Standards Mag.*, vol. 2, no. 2, pp. 62–68, Jun. 2018.

[4] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the Internet of Things and industry 4.0," *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 17–27, Mar. 2017.

[5] R. Salazar, T. Godfrey, L. Winkel, N. Finn, C. Powell, B. Rolfe, and M. Seewald, "Utility applications of time sensitive networking white paper (D3)," White Paper, Oct. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8870295

[6] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The time-triggered Ethernet (TTE) design," in *Proc. 8th IEEE Int. Symp. Object-Oriented Real-Time Distrib. Comput. (ISORC)*, May 2005, pp. 22–33.

[7] G. Carvajal, L. Araneda, A. Wolf, M. Figueroa, and S. Fischmeister, "Integrating dynamic-TDMA communication channels into COTS Ethernet networks," *IEEE Trans. Ind. Informat.*, vol. 12, no. 5, pp. 1806–1816, Oct. 2016.

[8] (Feb. 2020). *IEEE Audio and Video Bridging Task Group*. [Online]. Available: https://1.ieee802.org/tsn/

[9] *IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks—Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements*, IEEE Standard 802.1Qcc-2018 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018), Oct. 2018, pp. 1–208.

[10] A. Nasrallah, V. Balasubramanian, A. Thyagaturu, M. Reisslein, and H. ElBakoury, "Reconfiguration algorithms for high precision communications in time sensitive networks," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2019, pp. 1–6.

[11] R. Santos, R. Marau, A. Vieira, P. Pedreiras, A. Oliveira, and L. Almeida, "A synthesizable Ethernet switch with enhanced real-time features," in *Proc. 35th Annu. Conf. IEEE Ind. Electron.*, Nov. 2009, pp. 2817–2824.

[12] P. Pedreiras and L. Almeida, "The flexible time-triggered (FTT) paradigm: An approach to QoS management in distributed real-time systems," in *Proc. Int. Parallel Distrib. Process. Symp.*, Apr. 2003, p. 9.

[13] A. Gothard, R. Kreifeldt, and C. Turner, "AVB for automotive use white paper," AVnu Alliance, Beaverton, OR, USA, Tech. Rep., Oct. 2014. [Online]. Available: https://avnu.org/wp-content/uploads/2014/05/2014-11-20_AVnu-Automotive-White-Paper_Final_Approved.pdf

[14] *IEEE Standard for Local and Metropolitan Area Networks Time-Sensitive Networking Profile for Automotive In-Vehicle Ethernet Communications*, IEEE Standard P802.1DG/1.1, Oct. 2019.

[15] S. Varadarajan and T.-C. Chiueh, "EtheReal: A host-transparent real-time fast Ethernet switch," in *Proc. 6th Int. Conf. Netw. Protocols (ICNP)*, Austin, TX, USA, 1998, pp. 12–21, doi: 10.1109/ICNP.1998.723721.

[16] P. Meyer, T. Steinbach, F. Korf, and T. C. Schmidt, "Extending IEEE 802.1 AVB with time-triggered scheduling: A simulation study of the coexistence of synchronous and asynchronous traffic," in *Proc. IEEE Veh. Netw. Conf.*, Dec. 2013, pp. 47–54.

[17] J. Ko, J.-H. Lee, C. Park, and S.-K. Park, "Research on optimal bandwidth allocation for the scheduled traffic in IEEE 802.1 AVB," in *Proc. IEEE Int. Conf. Veh. Electron. Saf. (ICVES)*, Nov. 2015, pp. 31–35.

[18] H.-T. Lim, L. Völker, and D. Herrscher, "Challenges in a future IP/Ethernet-based in-car network for real-time applications," in *Proc. 48th Design Automat. Conf. (DAC)*, 2011, pp. 7–12.

[19] J. Migge, J. Villanueva, N. Navet, and M. Boyer, "Insights on the performance and configuration of AVB and TSN in automotive Ethernet networks," in *Proc. 9th Eur. Congr. Embedded Real Time Softw. Syst. (ERTS)*, Toulouse, France, Jan. 2018, pp. 1–10. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01746132

[20] R. Queck, "Analysis of Ethernet AVB for automotive networks using network calculus," in *Proc. IEEE Int. Conf. Veh. Electron. Saf. (ICVES)*, Jul. 2012, pp. 61–67.

[21] I. Alvarez, L. Almeida, and J. Proenza, "A first qualitative comparison of the admission control in FTT-SE, HaRTES and AVB," in *Proc. IEEE World Conf. Factory Commun. Syst. (WFCS)*, May 2016, pp. 1–4.

[22] *IEEE Standard for Local and Metropolitan Area Networks—Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*, IEEE Standard 802.1AS-2011, Mar. 2011, pp. 1–292.

[23] *IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams*, IEEE Standard 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005), Jan. 2009, pp. C1–72.

[24] *IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks Amendment 14: Stream Reservation Protocol (SRP)*, IEEE Standard 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005), Sep. 2010.

[25] *IEEE Standard for Local and Metropolitan Area Networks—Audio Video Bridging (AVB) Systems*, IEEE Standard 802.1BA-2011, Sep. 2011, pp. 1–45.

[26] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Standard 1588-2008 (Revision of IEEE Std 1588-2002), Jul. 2008, pp. 1–300.

[27] J. Cao, M. Ashjaei, P. J. L. Cuijpers, R. J. Bril, and J. J. Lukkien, "An independent yet efficient analysis of bandwidth reservation for credit-based shaping," in *Proc. 14th IEEE Int. Workshop Factory Commun. Syst. (WFCS)*, Jun. 2018, pp. 1–10.

[28] M. Ashjaei, L. Silva, M. Behnam, P. Pedreiras, R. J. Bril, L. Almeida, and T. Nolte, "Improved message forwarding for multi-hop HaRTES real-time Ethernet networks," *J. Signal Process. Syst.*, vol. 84, no. 1, pp. 47–67, Jul. 2016.

[29] J. K. Strosnider, J. P. Lehoczky, and L. Sha, "The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments," *IEEE Trans. Comput.*, vol. 44, no. 1, pp. 73–91, Jan. 1995.

[30] G. C. Buttazzo, "Periodic task scheduling," in *Hard Real-Time Computing Systems*. Boston, MA, USA: Springer, 2011, pp. 79–118, doi: 10.1007/978-1-4614-0676-1_4.

[31] L. Almeida, P. Pedreiras, and J. A. G. Fonseca, "The FTT-CAN protocol: Why and how," *IEEE Trans. Ind. Electron.*, vol. 49, no. 6, pp. 1189–1201, Dec. 2002.

[32] G. Bechtel, B. Gale, M. Kicherer, and D. Olsen, "Automotive Ethernet AVB functional and interoperability specification," AVnu Alliance, Beaverton, OR, USA, Tech. Rep., May 2015. [Online]. Available: https://avnu.org/wp-content/uploads/2014/05/AutoCDSFunctionalSpec-1_4-public_with_legal_notices.pdf

[33] L. L. Bello, "Novel trends in automotive networks: A perspective on Ethernet and the IEEE audio video bridging," in *Proc. IEEE Emerg. Technol. Factory Automat. (ETFA)*, Sep. 2014, pp. 1–8.
[34] A. Mehmed, W. Steiner, M. Antlanger, and S. Punnekkat, "System architecture and application-specific verification method for fault-tolerant automated driving systems," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2019, pp. 39–44.
[35] L. Moutinho, P. Pedreiras, and L. Almeida, "A real-time software defined networking framework for next-generation industrial networks," *IEEE Access*, vol. 7, pp. 164468–164479, 2019.

INÉS ÁLVAREZ (Student Member, IEEE) received the degree in computer science (computer engineering) and the master's degree in computer engineering from the University of the Balearic Islands (UIB), Palma de Mallorca, Spain, in 2014 and 2016, respectively, where she is currently pursuing the Ph.D. degree in information and communications technologies.

She is also a Lecturer with the Department of Mathematics and Computer Science, UIB. Her research interests include dependable and real-time systems, fault-tolerant distributed systems, and dependable communication topologies.

LUÍS MOUTINHO (Member, IEEE) was born in Aveiro, Portugal, in 1987. He received the M.Sc. degree in electronics and telecommunications engineering and the Ph.D. degree in telecommunications from the University of Aveiro, Portugal, in 2010 and 2019, respectively. He is currently an Invited Adjunct Professor with the Escola Superior de Tecnologia e Gestão de Águeda (ESTGA), Portugal. He is also a Research Assistant with the Instituto de Telecomunicações (IT), Portugal. He has published a book chapter and over 15 papers in conferences and journals related to his domains of interest, particularly real-time communications for industrial systems, software-defined networking, and vehicular networks. He was a member of the Organizing Committee of the 12th IEEE World Conference on Factory Communication Systems (WFCS), Portugal, in 2016, and the Workshop Chair with the 1st EAI International Conference on Future Intelligent Vehicular Technologies (Future5V), Portugal, in 2016.

PAULO PEDREIRAS (Senior Member, IEEE) graduated in electronics and telecommunications engineering, in 1997. He received the Ph.D. degree in electro technical engineering from the University of Aveiro, Portugal, in 2003. He is currently an Assistant Professor with the Electronics, Telecommunications and Informatics Department, University of Aveiro. He is also with the Portuguese Telecommunications Institute, Aveiro site, coordinating the Embedded Systems Group-AV. His current research interests include real-time embedded systems, wireless communications, electrical instrumentation, and industrial communications. He has participated more than 15 national and European research projects, with coordination responsibilities at diverse levels. Since 2000, he has been authored or coauthored over 150 papers in international peer-reviewed conferences and journals. He participates regularly on the technical program committees of some of the more relevant events of his research area, such as WFCS, SIES, and ETFA. He has participated on the organization with several international scientific events, such as the WFCS2015 Program Co-Chair, the WFCS2016 General Co-Chair, the ETFA2017-T2 Track Chair, WFCS2017, the WFCS2018 Steering Committee, and the ETFA2019 Workshop Co-Chair. He collaborates regularly, as a Reviewer, in several top international journals, such as the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, the *Journal of Systems Architecture* (Elsevier), and the *Real-Time Systems Journal* (Springer).

DANIEL BUJOSA received the degree in automation and industrial electronic engineering and the master's degree in industrial engineering from the University of the Balearic Islands (UIB), Palma de Mallorca, Spain, in 2017 and 2019, respectively. He is currently pursuing the Ph.D. degree with the School of Innovation, Design and Engineering, Mälardalen University.

His research interests include dependable and real-time systems, fault-tolerant distributed systems, and dependable communication topologies.

JULIÁN PROENZA (Senior Member, IEEE) received the Licenciatura en Ciencias Físicas degree in physics and the Ph.D. degree in informatics from the University of the Balearic Islands (UIB), Palma de Mallorca, Spain, in 1989 and 2007, respectively.

He is currently a Lecturer with the Department of Mathematics and Computer Science, UIB. His research interests include dependable and real-time systems, fault-tolerant distributed systems, adaptive systems, clock synchronization, dependable communication topologies, and field-bus and industrial networks.

Dr. Proenza has been a member of the IEEE Industrial Electronics Society (IES), since 2009. He is also a member of the IES Technical Committee on Factory Automation.

LUIS ALMEIDA (Senior Member, IEEE) graduated in electronics and telecommunications engineering from the University of Aveiro, Portugal, in 1988. He received the Ph.D. degree in electrical engineering from the University of Aveiro, in 1999. He is currently an Associate Professor with the Electrical and Computer Engineering Department, University of Porto (UP), Portugal, where he coordinates the Distributed and Real-time Embedded Systems Laboratory (DaRTES). He is also the Vice-Director of the CISTER Research Center on Real-Time and Embedded Computing Systems. He has published over 300 papers in related conferences and journals. He regularly participates with the organization of scientific events in his domains of interest. His research interests include real-time communications for distributed industrial/embedded systems, for teams of cooperating agents and for sensor networks. Trustee of the RoboCup Federation, from 2008 to 2016, and the Vice-President, from 2011 to 2013. He is the Vice-Chair of the IEEE Technical Committee on Real-Time Systems. He was a Program and the General Chair of the IEEE Real-Time Systems Symposium, in 2011 and 2012, respectively. He serves as an Associate Editor for the *Journal of Real-Time Systems* (Springer), the *Journal of Systems Architecture* (Elsevier), and the *International Journal on Advanced Robotic Systems* (SAGE).